



**CENTRO UNIVERSITARIO TECNOLÓGICO CEUTEC**

**TEGUCIGALPA, M.D.C.**

**ASIGNATURA:**

**Desarrollo Web I**

**DOCENTE:**

**Ing. Jonie Miralda**

**Actividad en clase**

**Consumo de API de terceros**

**Presentado por:**

Grupo	# CUENTA:
Paola Gabriela Rodríguez	T32421205
Gerardo Antonio Perez	T32421242
Carlos Francisco Garcia	T32351150

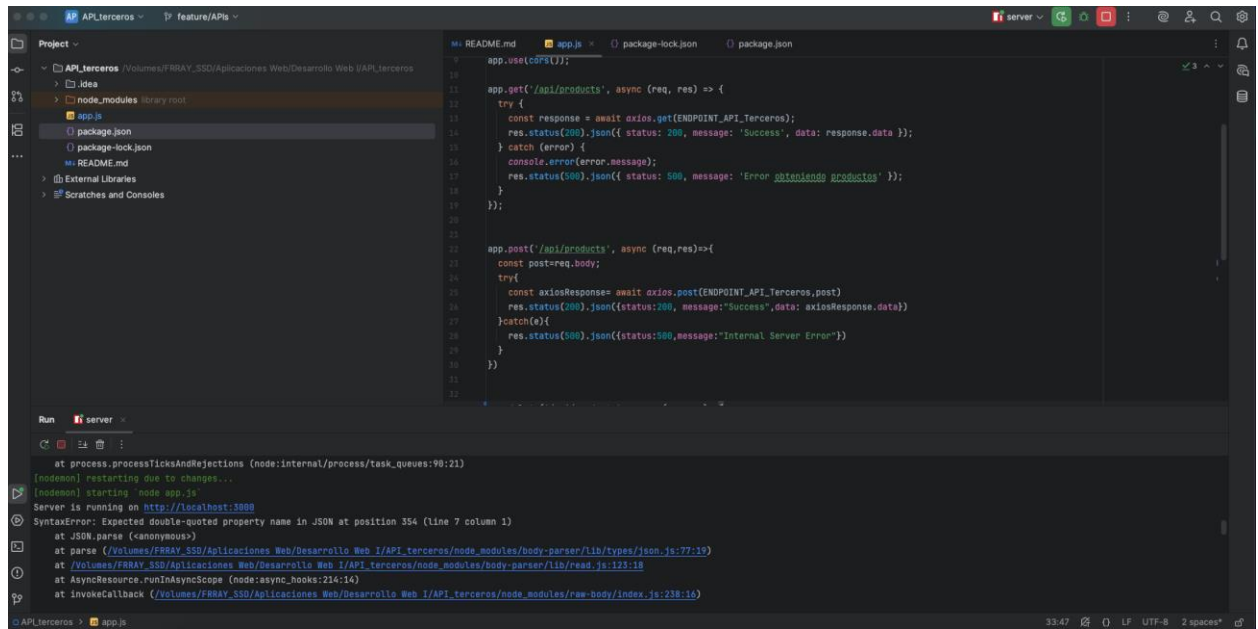
20 DE JUNIO DE 2025

GET

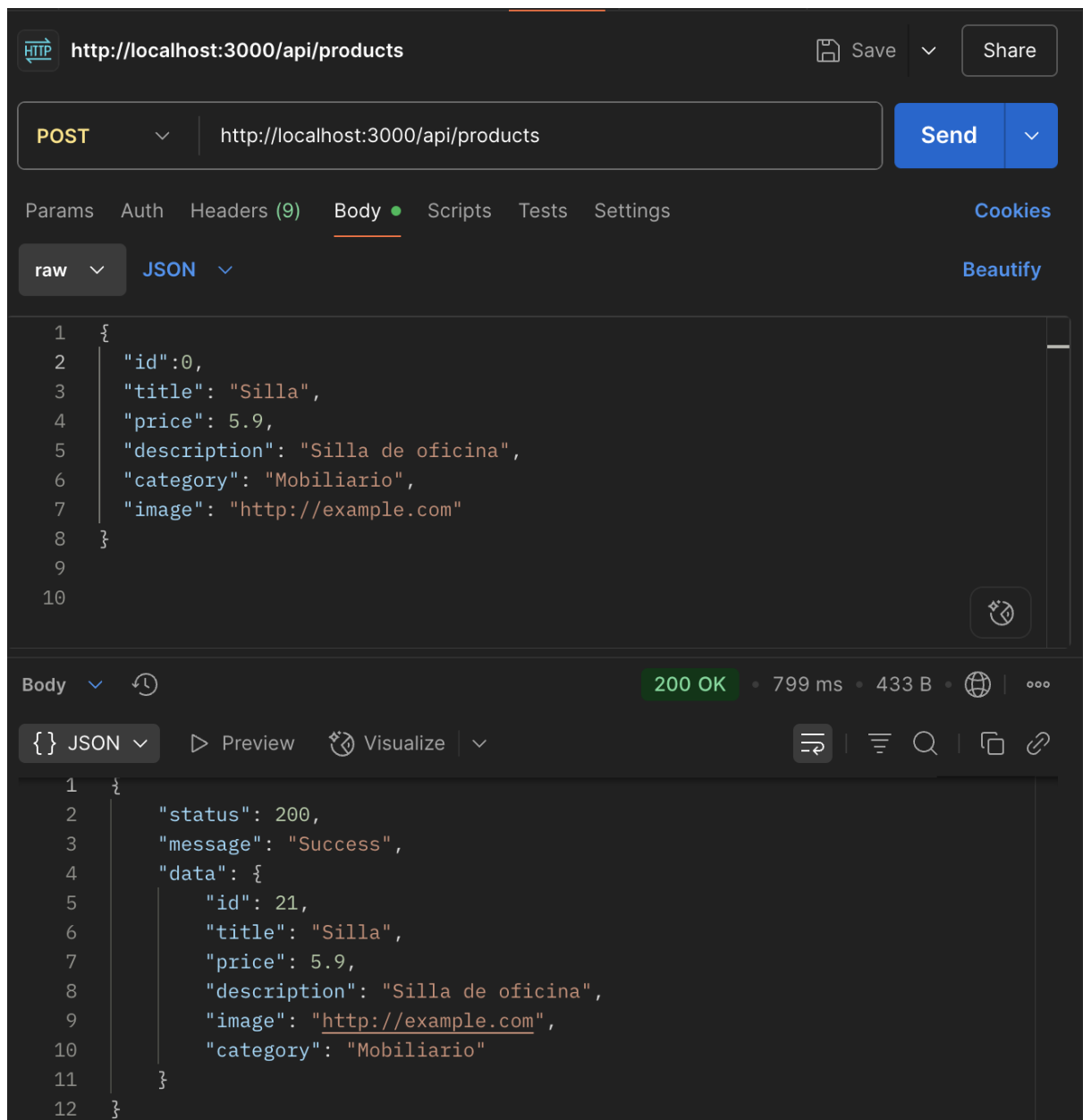
The image shows a development environment with VS Code and Postman. In VS Code, the file explorer on the left shows a project structure with 'API\_Terceros' and 'app.js'. The 'app.js' file is open in the editor, showing a Node.js Express application that uses axios to fetch data from 'https://fakestoreapi.com/products'. The terminal at the bottom shows the command 'npm install express axios' and the output indicating successful installation of 78 packages.

Below the VS Code window, the Postman interface is shown. The URL bar is set to 'http://localhost:3000/api/products'. The 'GET' method is selected, and the 'Send' button is visible. The 'Body' tab is active, showing the response in JSON format. The response status is '200 OK', and the body contains a single product object:

```
{
  "status": 200,
  "message": "Success",
  "data": [
    {
      "id": 1,
      "title": "Fjallraven - Foldsack No. 1 Backpack, Fits 15 Laptops",
      "price": 109.95,
      "description": "Your perfect pack for everyday use and walks in the forest. Stash your laptop (up to 15 inches) in the padded sleeve, your everyday",
      "category": "men's clothing",
      "image": "https://fakestoreapi.com/img/81fPKd-2AYL._AC_SL1500_.jpg"
    }
  ]
}
```



## PUT



HTTP **PUT** `http://localhost:3000/api/products` Save Share

**POST** `http://localhost:3000/api/products` Send

Params Auth Headers (9) **Body** Scripts Tests Settings Cookies

raw **JSON** Beautify

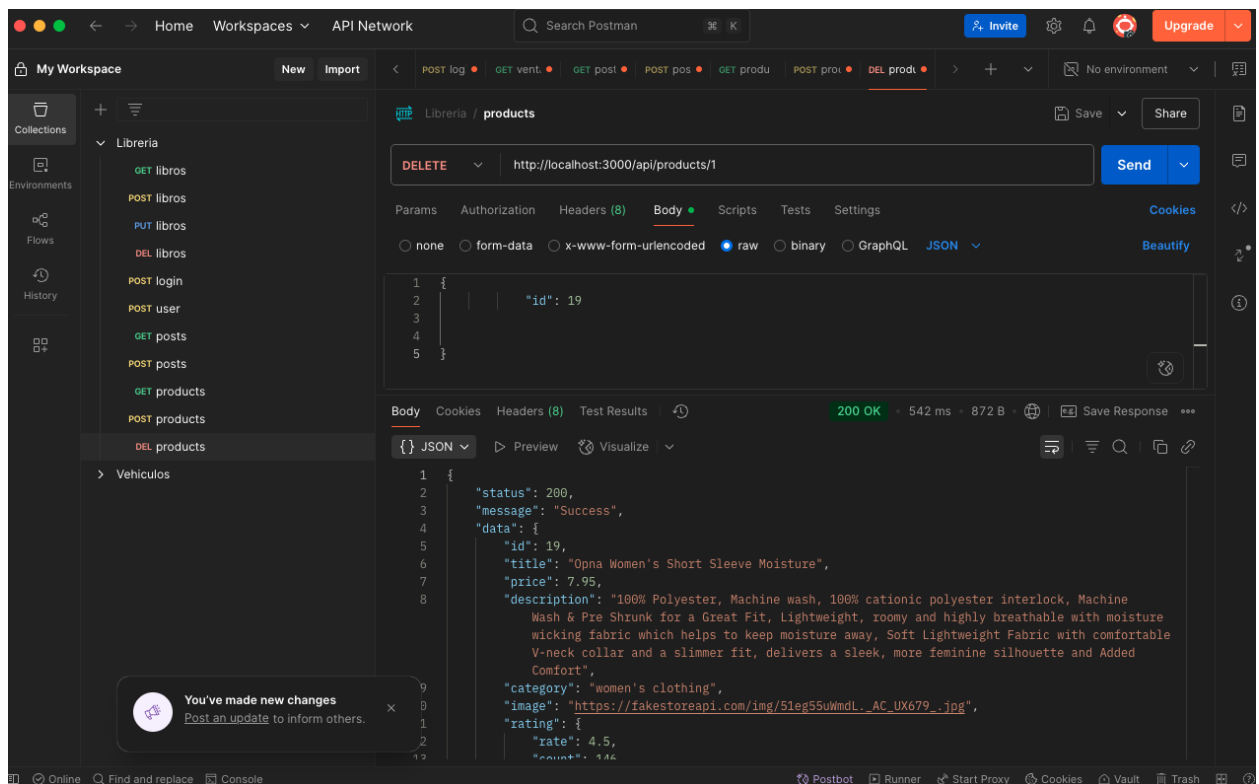
```
1 {
2   "id": 0,
3   "title": "Silla",
4   "price": 5.9,
5   "description": "Silla de oficina",
6   "category": "Mobiliario",
7   "image": "http://example.com"
8 }
9
10
```

Body 200 OK • 799 ms • 433 B • 🌐 ⋮

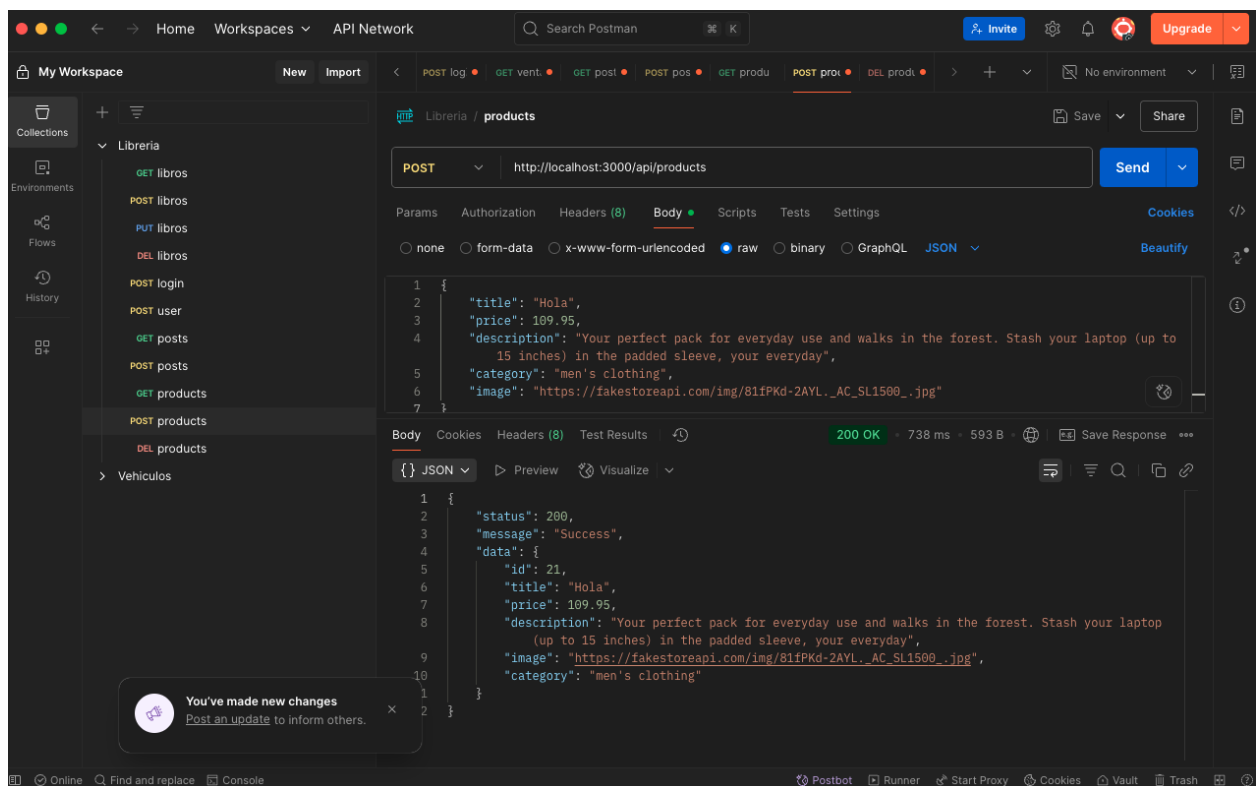
**{}** JSON Preview Visualize ⋮

```
1 {
2   "status": 200,
3   "message": "Success",
4   "data": {
5     "id": 21,
6     "title": "Silla",
7     "price": 5.9,
8     "description": "Silla de oficina",
9     "image": "http://example.com",
10    "category": "Mobiliario"
11  }
12 }
```

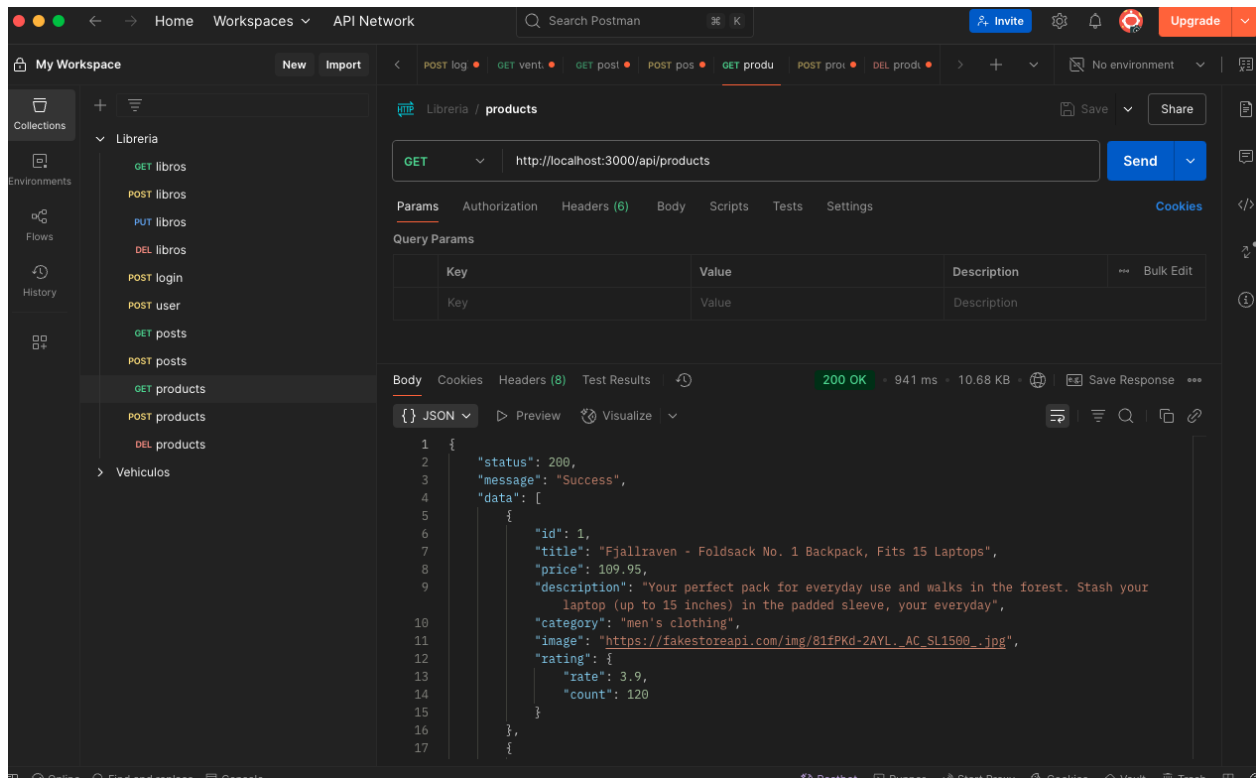
## DELETE



## POST



## GET



### Explicación de los métodos y funciones utilizadas en tu implementación.

En nuestra implementación grupal utilizamos Express para configurar el servidor y definir las rutas necesarias para las operaciones CRUD. Usamos métodos como `app.get`, `app.post`, `app.put` y `app.delete` para manejar las distintas solicitudes del cliente. También integramos la librería `axios` para realizar peticiones HTTP a la API externa de FakeStoreAPI, lo que nos permitió obtener productos (GET), crear nuevos (POST), actualizarlos (PUT) y eliminarlos (DELETE). A través de `req.body` y `req.params` accedimos a los datos enviados por el cliente, como los campos del producto o su ID. Además, configuramos `express.json()` para procesar datos en formato JSON. Esta práctica nos ayudó a comprender mejor cómo consumir una API de terceros y construir un servicio backend funcional en equipo.

### **Desafíos encontrados y cómo los resolviste.**

Mi desafío principal fue integrar Axios para hacer la solicitud GET a la API. Al principio no lograba obtener los datos correctamente y recibía errores. Para solucionarlo, revisé la documentación y probé el endpoint con Postman. Así entendí mejor cómo usar `axios.get()` y pude obtener y mostrar los productos correctamente.

Uno de los desafíos es el método Delete, ya que no estaba seguro de si solo se necesitaba el id o el objeto completo, ya que la documentación de la página no es muy clara, además para el método delete es necesario aplicar una string template lo cual se tuvo que investigar